

1. [On Factoring Polynomials](#)
2. [The Lindsey-Fox Algorithm for Factoring Polynomials](#)
3. [Timing of the Lindsey-Fox Algorithm](#)
4. [References for the LF Algorithm](#)
5. [More Details on the Third Stage of the LF Algorithm](#)
6. [Mathematical Principles for the LF Algorithm](#)
7. [Zero location for Polynomials with Random Coefficients](#)

On Factoring Polynomials

Factoring a high degree polynomial has been considered a difficult problem since the beginning of mathematics. We describe the strategies used to attack this problem. The results are from a group called the "Polynomial Club" (Jim Fox, Sidney Burrus, Gary Sitton, and Sven Treitel)

Introduction

Polynomials form one of the oldest classes of mathematical functions in mathematics with an important history in science, engineering, and other quantitative fields [[Pan, 1997](#)], [[Pan, 1998](#)], [[Traub, 1998](#)], [[Derbyshire, 2006](#)], [[Sitton, 2003](#)]. Part of the polynomial's appeal comes from the fact that it may be numerically evaluated using a finite number of multiplications and additions. Another advantage it presents is its use in modeling physical processes or complicated mathematical functions. Polynomials are used to build expansion systems or basis sets in various vector spaces. Segments of polynomials are used to create splines. It is worth studying the basic ideas and the variety of applications.

Polynomials are introduced early in the teaching of algebra as a means of demonstrating basic principles and methods, e.g. substitution, simplification, and factoring. Most people are aware that there exist formulas for factoring quadratic (2nd degree), cubic (3rd degree), and quartic (4th degree) polynomials. In 1824 the mathematician Niels Abel proved that there is no possible "closed form" solution in terms of basic operations for the general pentic or quintic (5th degree) polynomial or those of higher degrees. This fact requires the development of effective numerical methods for iteratively factoring polynomials above degree 4.

The results reported in these modules are from a group at Rice University called the "Polynomial Club" (Jim Fox, Sidney Burrus, Gary Sitton, and Sven Treitel). The program was designed and written by Jim Fox.

Basic Definitions

The definition of an N th degree polynomial is

Equation:

$$f(z) = a_0 + a_1z + a_2z^2 + \cdots + a_Nz^N = \sum_{k=0}^N a_kz^k$$

where both $f(z)$ and z are complex valued and the coefficients a_k can be complex but are often real valued. N , which is the highest power of z in the polynomial, is called the degree (or sometimes the order) of the polynomial. The number of coefficients is $N + 1$.

From the various forms of the **Fundamental Theorem of Algebra**, one can show that all polynomials can also be expressed in a “factored” form by **Equation:**

$$f(z) = a_N \prod_{r=1}^N (z - z_r)$$

where the possibly complex valued z_r are called the “zeros” of the polynomial since $f(z_k) = 0$. Because the zeros are not necessarily distinct, a unique form of [\[link\]](#) is given by

Equation:

$$f(z) = a_N \prod_{k=1}^L (z - z_k)^{M_k}$$

where M_k is the multiplicity of the k^{th} zero, L is the number of distinct zeros, and $\sum_k M_k = N$. The first degree polynomials $(z - z_k)$ in [\[link\]](#) and [\[link\]](#) are called the factors of $f(z)$. Note the analogy between polynomials and integers. Indeed, multiplying two polynomials is the same operation as multiplying two integers (except for carrying) or convolving two number sequences.

Creating [\[link\]](#) from [\[link\]](#) or [\[link\]](#) is fairly straight forward and requires only a finite number of arithmetic operations, but finding [\[link\]](#) or [\[link\]](#) from [\[link\]](#) is difficult. That process is called “factoring” the polynomial

and is the topic of these notes. Actually, the effects of finite precision arithmetic sometimes make the "unfactoring" process of calculating [\[link\]](#) from [\[link\]](#) or [\[link\]](#) poorly defined because it depends on the sequence order of combining the zeros.

One can look at the zeros of a polynomial as being a second parameterization of the polynomial with the coefficient form of [\[link\]](#) being the first. A combination would be expressing an even degree polynomial as the product of two $N/2$ degree polynomials. Factoring and "unfactoring" can create a variety of parameterizations for both discrete-time signals and systems. It should be noted that the coefficients of the polynomial are nonlinear functions of all of the zeros, and the zeros of the polynomial are nonlinear functions of all of the coefficients. In some cases, a very small change in one zero can cause a huge change in the coefficients and/or a very small change in a coefficient can cause a huge change in the zeros. In these cases, the polynomials are called "poorly conditioned" and are very difficult to factor by any means.

The Four Problems for Polynomials

1. Evaluating the polynomial and its derivatives. "polyval" in Matlab or [Horner's method](#)
2. Factoring the polynomial. "roots" in Matlab
3. Composing the factors. "poly" in Matlab or convolution perhaps using the FFT.
4. Deflating the polynomial by a root. [Horner's method](#) or FFT division.

If the polynomial is represented in factored form [\[link\]](#), then the coefficient sum form [\[link\]](#) may be easily found by computing the product of the M linear polynomial factors. This is done by multiplying the linear factors one at a time and collecting all coefficients of like powers in the product. As will be shown later, this is equivalent to a cascaded discrete **convolution** of all of the polynomial coefficients where the r^{th} linear polynomial is represented as a doublet set of two coefficients $\{-z_r, 1\}$ where the leading coefficient $-z_r$ is usually complex. This computation allows the reconstruction of the coefficients of the polynomial corresponding to this root set. We call this process **unfactoring** but it can lead to large numerical

inaccuracies for a surprisingly small number of terms. The main concern here is with the inverse or factoring problem: given the coefficient form [\[link\]](#), find the factored form [\[link\]](#).

The common case of purely real polynomial coefficients leads to a simplification: All of the complex roots of a **real** polynomial occur in complex **conjugate** pairs. Given the r^{th} complex root $z_r = x_r + iy_r$, its complex conjugate is given by $z_r^* = x_r - iy_r$.

Thus for $y_r \neq 0$, a complex root z_r will always be associated with its conjugate z_r^* to form a conjugate root pair. This pair of roots represents the product of two linear factors forming a real quadratic or second degree factor:

Equation:

$$(z - z_r)(z - z_r^*) = (x^2 + y^2) - 2x_r z + z^2$$

From [\[link\]](#) a polynomial may be defined as the product of all of its factors. If the factors have only real coefficients, then the product of all the factors will have only real coefficients a_n . Thus only half of the complex roots need be found, say in the upper complex half-plane with positive imaginary parts, i.e. $y_r \geq 0$. The associated conjugate roots in the lower half-plane may be trivially derived by simple negation of their imaginary parts.

Most of the results in these modules are based on extensive numerical experimentation. We have built on existing theory and techniques with empirically derived rules and algorithms that perform well on well-conditioned polynomials and, in many cases, specifically applied to signal processing applications with random coefficients.

Factoring Polynomials

Below are three approaches to factoring polynomials:

- **Find and Deflate:** The usual algorithms for factoring polynomials start by guessing or estimating the value of a zero, then using some

descent method on $|f(z)|$, find a single zero. This zero is then removed from $f(z)$ by "deflation" ($f(z)$ is divided by the factor represented by the zero) and the process repeated on the reduced degree quotient polynomial. The descent method is often [Newton's](#) method which is implemented with [Horner's method](#). There may be problems with error accumulation which, after several deflations, results in failure. Also, if the zeros are not found and removed in a carefully chosen order, the quotient polynomial becomes poorly conditioned. This set of procedures is often simply called [Newton's](#) method. Methods presented in these notes include "Random Argument", "Chosen Argument", and "Pre-Whitening", all of which try to maintain or improve the conditioning of the quotient polynomial produced by deflation. The deflation itself must be done in a "stable" way to prevent error accumulation. In some cases, this involves using [Horner's method](#) and in others, the DFT.

- **Eigenvalue Method:** If the companion matrix for the polynomial is created, its eigenvalues are the zeros of the polynomial. Because very sophisticated algorithms have been developed for finding eigenvalues, this is a powerful and robust approach. However, it requires considerable computer memory for the matrices and is computationally inefficient and, therefore, slow. Matlab uses this approach.
- **Grid Search:** If one has knowledge from the structure of the polynomial what regions in the complex plane contain the zeros, a direct search can be used. Because a large class of polynomials, including those with random coefficients, have their zeros clusters around the unit circle, a very efficient polar coordinate grid search can be conducted to find good estimates for the zeros which are then found accurately by a Newton's or Laguerre's algorithm [18,19]. The Lindsey-Fox algorithm uses this approach. A description of [the Lindsey-Fox program](#)

Bibliography

1. J. Derbyshire, *Unknown Quantity: A Real and Imaginary History of Algebra*. Joseph Henry Press, an imprint of the National Academies Press, Washington, 2006.

2. V. Y. Pan, "Solving a polynomial equation: some history and recent progress". *SIAM Review*. No. 2, Vol. 39, June 1997. pp. 187–220.
3. V. Y. Pan, "Solving Polynomials with Computers". *American Scientist*. No. 1, Vol. 86, January-February 1998. pp. 62–69.
4. G. Sitton, C. S. Burrus, J. W. Fox, and S. Treitel, "Factoring Very High Degree Polynomials in Signal Processing". *Signal Processing Magazine*. No. 6, Vol. 20, November 2003. pp. 27–42.
5. J. F. Traub, "Another Algorithm". *American Scientist*. No. 2, Vol. 86, March-April 1998. pp. 108–109.
6. J. M. McNamee "A Bibliography on Roots of Polynomials", *J. Comput. Appl. Math.*, 47 (1993) pp. 391-394. 78 (1997). 110 (1999) pp. 305-306.

The Lindsey-Fox Algorithm for Factoring Polynomials

The program which implements the Lindsey-Fox algorithm that factors high degree polynomial is organized in three stages. Jim Fox is the primary author and architect of the program.

The Lindsey-Fox algorithm uses the FFT (fast Fourier transform) to very efficiently conduct a grid search in the complex plane to find accurate approximations to the N roots (zeros) of an N th degree polynomial. The power of this grid search allows a new polynomial factoring strategy that has proven to be very effective for a certain class of polynomials.

This algorithm was conceived of by Pat Lindsey and implemented by Jim Fox in a package of computer programs created to factor high-degree polynomials. It was originally designed and has been further developed to be particularly suited to [polynomials](#) with real, random coefficients. In that form, it has proven to be very successful by factoring thousands of polynomials of degrees from one thousand to hundreds of thousand as well as several of degree one million and one each of degree two million and four million. In addition to handling very high degree polynomials, it is accurate, fast, uses minimum memory, and is programmed in the widely available language, Matlab. There are practical applications, often cases where the coefficients are samples of some natural signal such as speech or seismic signals, where the algorithm is appropriate and useful. However, it is certainly possible to create special, ill-conditioned polynomials that it cannot factor, even low degree ones. The basic ideas of the algorithm were first published by Lindsey and Fox in 1992 [1] and reprinted in 1996 [2]. After further development, another paper was published in 2003 [3]. The program was made available to the public in March 2004 on the Rice University web site [14]. A more robust version-2 was released in March 2006 and updated later in the year. The references are in a separate [module](#).

The Three Stages of the Lindsey-Fox Program

The strategy implemented in the Lindsey-Fox algorithm to factor polynomials is organized in three stages. The first evaluates the polynomial over a grid on the complex plane and conducts a direct search for potential zeros. The second stage takes these potential zeros and “polishes” them by

applying Laguerre's method [18,19] to bring them close to the actual zeros of the polynomial. The third stage multiplies these zeros together or "unfactors" them to create a polynomial that is verified against the original. If some of the zeros were not found, the original polynomial is "deflated" by dividing it by the polynomial created from the found zeros. This quotient polynomial will generally be of low order and can be factored by conventional methods with the additional, new zeros added to the set of those first found. If there are still missing zeros, the deflation is carried out until all are found or the whole program needs to be restarted with a finer grid. This system has proven to be fast, accurate, and robust on the class of polynomials with real, random coefficients and other similar, well-conditioned polynomials.

The Lindsey-Fox program can be outlined by:

Stage one: the grid search for prospective zeros

1. Construct a polar coordinate grid on the complex plane with spacing derived from the degree of the polynomial being factored
2. Use the FFT to evaluate the polynomial at each node along the concentric circles of the grid.
3. Search over each 3x3 set of values for relative minima. If the center value is less than the edge values, it is a prospective zero by the Minimum Modulus Theorem of complex analysis.

Stage two: polish the prospective zeros

1. Apply Laguerre's algorithm to each prospective zero, correcting it to a better approximation of the "true" zero of the polynomial
2. Test the set of polished zeros for uniqueness and discard any duplicates to give a set of candidate zeros

Stage three: Unfactor, perhaps deflate, and verify

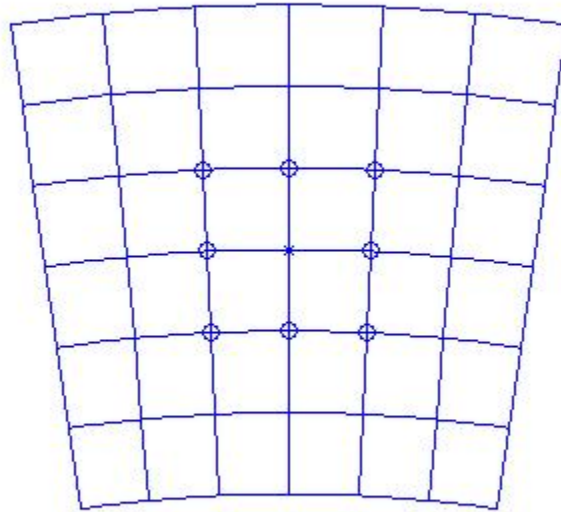
1. Unfactor the polished zeros i.e., reconstruct a candidate polynomial in coefficient form from the polished candidate zeros

2. If the degree of the reconstructed polynomial is the same as that of the original polynomial and differences in their coefficients are small, the factoring is successful and finished
3. If some zeros were missed, deflate and factor the quotient polynomial. If that does not find all of the missed zeros, deflate and factor again until all are found or until no new ones are found
4. If deflation finds all the zeros that it can, and it still has not found them all, design a new grid with a finer spacing and return to stage one. If four restarts do not find them all and/or the reconstruction error is not small, declare failure.

Description of the three stages

Stage one is the reason this algorithm is so efficient and is what sets it apart from most other factoring algorithms. Because the FFT (fast Fourier transform) is used to evaluate the polynomial, a fast evaluation over a dense grid in the complex plane is possible. In order to use the FFT, the grid is structured in polar coordinates. In the first phase of this stage, a grid is designed with concentric circles of a particular radius intersected by a set of radial lines. The positions and spacing of the radial lines and the circles are chosen to give a grid that will hopefully separate the expected roots. Because the zeros of a polynomial with random coefficients have a fairly uniform angular distribution and are clustered close to the unit circle, it is possible to design an evaluation grid that fits the expected root density very well. In the second phase of this stage, the polynomial is evaluated at the nodes of the grid using the fast Fourier transform (FFT). It is because of the extraordinary efficiency and accuracy of the FFT that a direct evaluation is possible. In the third phase of this first stage, a search is conducted over all of the 3 by 3 node cells formed in the grid. For each 3 by 3 cell (see Figure below), if the value of the polynomial at the center node of the cell (the "x") is less than the values at all 8 of the nodes on the edges of the cell (the "o's"), the center is designated a candidate zero. This rule is based on the "Minimum Modulus Theorem" which states that if a relative minimum of the absolute value of an analytic function over an open region exists, the minimum must be a zero of the function. Finally, this set of prospective zeros is passed to the second stage. The number is usually slightly larger

than the degree because some were found twice or mistakes were made. The number could be less if some zeros were missed.



Section of the polar coordinate grid showing a 3 node by 3 node cell

Stage two is more traditional than the other two. It “polishes” each of the prospective zeros found by the grid search. The first phase consists of applying an iterative algorithm to improve the accuracy of the location found by the grid search. In earlier versions of the program, Newton’s method was used but analysis and experiment showed that Laguerre’s method [18,19] was both more robust and more accurate. Even though it required more calculation than Newton’s method for each iteration, it converged in fewer iterations. The second phase of the second stage checks

for duplications. A “fuzzy” uniqueness test is applied to each zero to eliminate any cases where on two or more prospective zeros, iterations converged to the same zero. If the number of unique, polished zeros is less than the degree of the polynomial, deflation later will be necessary. If the number is greater, some error has occurred. This stage consumes the largest part of the execution time of the total factorization, but it is crucial to the final accuracy of the roots. One of the two criteria for success in factoring a polynomial is that each root must have been successfully polished against the original polynomial.

Stage three has several phases and possible iterations or even restarting. The first phase of the third stage takes all of the unique, polished zeros that were found in the first two stages and multiplies them together into the coefficient form of a candidate polynomial (“unfactors” the zeros). If the degree of this reconstructed polynomial is the same as that of the original polynomial and if the difference in their coefficients is small, the factorization is considered successful. Often, however, several zeros were missed by the grid search and polish processes of stage one and two, or the uniqueness test discarded a legitimate zero (perhaps it is multiple), so the original polynomial is “deflated” (divided) by the reconstructed polynomial and the resulting (low degree) quotient is factored for the missing zeros. If that doesn’t find them all, the deflation process is repeated until they are all found. This allows the finding of multiple roots (or very tightly clustered roots), even if some of them were discarded earlier. If, in the unusual case, these further iterations of deflation do not find all of the missing zeros, a new, finer grid is constructed and the whole process started again at stage one. More details on the third stage are in another [module](#).

Multiple order and clustered roots are unusual in random coefficient polynomials. But, if they happen or if factoring an ill-conditioned polynomial is attempted, the roots will be found with the Lindsey-Fox program in most cases but with reduced accuracy. If there are multiple order zeros (M th order with M not too high), the grid search will find them, but with multiplicity one. The polishing will converge to the multiple order root but not as fast as to a distinct root. In the case of a cluster with Q zeros that fall within a single cell, they are erroneously identified as a single zero and the polishing will converge to only one of them. In some cases, two zeros

can be close to each other in adjacent cells and polish to the same point. In all of these cases, after the uniqueness test and deflation, the quotient polynomial will contain a $M-1$ order zero and/or $Q-1$ zeros clustered together. Each of these zeros will be found after $M-1$ or $Q-1$ deflations. There can be problems here because Laguerre's polishing algorithm is not as accurate and does not converge as fast for a multiple zero and it may even diverge when applied to tightly clustered zeros. Also, the condition of the quotient polynomial will be poorer when multiple and clustered zeros are involved. If multiple order zeros are extremely far from the unit circle, the special methods for handling multiple roots developed by Zhonggang Zeng [5] are used. Zeng's method is powerful but slow, and hence only used in special cases [6]. [References](#)

Successful completion of the factoring of a polynomial requires matching zeros on the complex plane measured by the convergence of Laguerre's algorithm on each of the zeros. It also requires matching the polynomial reconstructed from the found zeros with the original polynomial by measuring the maximum difference in each coefficient.

Characteristics of the Lindsey-Fox Algorithm

Because the FFT is such an efficient means of evaluating the polynomial, a very fine grid can be used which will separate all or almost all of the zeros in a reasonable time. And because of the fineness of the grid, the starting point is close to the actual zero and the polishing almost always converges in a small number of steps (convergence is often a serious problem in traditional approaches). And because the searching and polishing is done on the original polynomial, they can be done on each root simultaneously on a parallel architecture computer. Because the searching is done on a 3 by 3 cell of the grid, no more than three concentric circles of the grid need be kept in memory at a time, i.e., it is not necessary to have the entire grid in memory. And, some parallelization of the FFT calculations can be done.

Deflation is often a major source of error or failure in a traditional iterative algorithm. Here, because of the good starting points and the powerful polisher, very few stages of deflation are generally needed and they produce

a low order quotient polynomial that is generally well-conditioned. Moreover, to control error, the unfactoring (multiplying the found roots together) is done in the FFT domain (for degree larger than 500) and the deflation is done partly in the FFT domain and partly in the coefficient domain, depending on a combination of stability, error accumulation, and speed factors.

For random coefficient polynomials, the number of zeros missed by the grid search and polish stages ranges from 0 to 10 or occasionally more. In factoring one 2 million degree polynomial, the search and polish stages found all 2 million zeros in one grid search and required no deflation which shows the power of the grid search on this class of polynomial. When deflation is needed, one pass is almost always sufficient. However, if you have a multiple zero or two (or more) very, very closely spaced zeros, the uniqueness test will discard a legitimate zero but it will be found by later deflation. Stage three has enough tests and alternatives to handle almost all possible conditions. But, by the very definition of random coefficients, it is hard to absolutely guarantee success.

The timings of the The Lindsey-Fox program can be found [here](#) and examples of root distributions of polynomials with random coefficients are [here](#).

Timing of the Lindsey-Fox Algorithm

Timing of the factoring of polynomials with the Lindsey-Fox algorithm.

Timings

A large number of polynomials with random coefficients were factored with the [Lindsey-Fox](#) program on a 2.6 MHz Pentium with 1 GB RAM in August 2003. The following table shows the average time required for different polynomial degrees. The first column is the polynomial degree (length minus one), the second column is the time in seconds required by Matlab using "roots", and the third column is the time in seconds using the Lindsey-Fox program "lroots" written in Matlab [14]. The polynomial coefficients were random numbers generated by Matlab.

Polynomial Degree	Time using roots()	Time using lroots()
50	0.004	0.04
100	0.020	0.06
200	0.140	0.10
500	3.110	0.23
1,000	24.750	0.50
2,000	250.740	1.20
5,000	13,891.000	6.34
6,000	"Out of memory"	7.49

10,000	"Out of memory"	21.45
100,000	"Out of memory"	1,769.00
150,000	"Out of memory"	4,822.00
250,000	"Out of memory"	9,875.00
500,000	"Out of memory"	45,574.00
1,000,000	"Out of memory"	353,848.00

In order to better understand the Lindsey-Fox program, the individual times required by the three stages algorithm factoring a 2,080,000 degree random coefficient polynomial on a 3 GHz Pentium with 4 GB RAM (run on 1/6/2006) was measured and is presented in the table below. Note the efficient search stage, the time consuming second stage (which can be easily parallelized), and the moderately demanding third stage which can be partially parallelized and which is not always needed.

Operation	Time in seconds	Time in days	Percent of total run time
Grid search	16,692	0.2	3.2%
Polish	295,388	3.4	58.0%
Unfactor & Deflate	196,675	2.3	38.8%

The references for the Lindsey-Fox algorithm can be found [here](#)

References for the LF Algorithm

References for the Lindsey-Fox polynomial factoring algorithm

References

Below are several references on the [Lindsey-Fox algorithm](#) and on the zero distribution for polynomials with random coefficients.

1. J. P. Lindsey and James W. Fox. "A method of factoring long z-transform polynomials", *Computational Methods in Geosciences*, SIAM, pp. 78-90, 1992.
2. Osman Osman (editor), *Seismic Source Signature Estimation and Measurement*, Geophysics Reprint Series #18, Society of Exploration Geophysicists (SEG), 1996, pp. 712-724.
3. Gary A. Sitton, C. Sidney Burrus, James W. Fox, and Sven Treitel. "Factoring very high degree polynomials". *IEEE Signal Processing Magazine*, 20(6):27-42, November 2003.
4. C. S. Burrus, J. W. Fox, G. A. Sitton, and S. Treitel, "Factoring High Degree Polynomials in Signal Processing", *Proceedings of the IEEE DSP Workshop*, Taos, NM, Aug. 3, 2004, pp. 156-157.
5. Zhonggang Zeng, "Computing Multiple Roots of Inexact Polynomials", ACM ISSAC, 2003. also: *Math. Comp.* 74 (2005), 869 - 903.
6. Zhonggang Zeng, Northeastern Illinois University, March 10, 2006, MultRoot -- A Matlab package computing polynomial roots and multiplicities <http://www.neiu.edu/~zzeng/multroot.htm>,
7. L. Arnold, "Über die nullstellenverteilung zufälliger polynome," *Mathematische Zeitschrift*, vol. 92, pp. 12–18, 1966.
8. Larry A. Shepp and Robert J. Vanderbei, "The Complex Zeros of Random Polynomials", *Transactions of the American Mathematical Society*, Vol 347, # 11, Nov. 1995, pp 4365-4384.
9. Ildar Ibragimov & Ofer Zeitouni, "On Roots of Random Polynomials", *Transactions of the American Mathematical Society*, vol 349, # 6, June 1997, pp 2427-2441.
10. Bharucha-Reid and Sambandham, *Random Polynomials*, Academic Press, 1986.

11. J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Prentice-Hall, 1963.
12. N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 1996. Second edition 2002. Chapter 5 on polynomials.
13. Lloyd N. Trefethen and David Bau, *Numerical Linear Algebra*, SIAM, 1997.
14. C. S. Burrus, J. W. Fox, G. A. Sitton, and S. Treitel, "Factoring Very High Degree Polynomials", *Rice Web Site*, March 10, 2006.
<http://www-dsp.rice.edu/software/fvhdp.shtml>
15. J. B. Moore, "A Convergent Algorithm for Solving Polynomial Equations", *Journal of the ACM*, 14(2):311-315, April 1967.
16. J. B. Moore, "A Consistently Rapid Algorithm for Solving Polynomial Equations", *Journal of the Institute of Mathematics and Its Applications*, 17:990-119, 1976.
17. M. A. Jenkins and S. F. Traub, "A Three-Stage Algorithm for Real Polynomials using Quadratic Iterations", *SIAM Journal on Numerical Analysis*, 545-566, 1970.
18. Laguerre's Method from Wikipedia:
http://en.wikipedia.org/wiki/Laguerre%27s_method
19. H. J. Orchard, "The Laguerre Method for Finding the Zeros of Polynomials", *IEEE Transaction on Circuits and Systems*, vol. 36, November 1989, pages 1377-1381.
20. C. P. Hughes and A. Nikeghbali, "The Zeros of Random Polynomials Cluster Uniformly near the Unit Circle", Report of the American Institute of Mathematics in Palo Alto and the Laboratoire de Probabilités et Modèles Aléatoires in Paris, prepublication no. 922, UMR 7599, June 2004. New version in arXiv:math/0406376v3, June 2007.

More details on the third stage of the Lindsey-Fox program can be found [here](#)

More Details on the Third Stage of the LF Algorithm

The third stage of the Lindsey-Fox algorithm multiplies the factors back together to check with the original polynomial coefficients.

Addendum: More details of the third (unfactor and deflation) stage

The **deflation stage** of the [Lindsey-Fox](#) program first separates the unique, polished candidate roots that were found by the grid search and polish stages into two sets, one, a set very, very near the unit circle and the other, a set containing the rest of them. Both groups are unfactored in the frequency domain by the rootSpectrum program. It takes each zero or root, r , forms the associated first degree factor, $(z - r)$, and pads with zeros to a length which is an appropriate power of 2; i.e. $[1, -r, 0, 0, 0, \dots]$. The DFT's of these terms are computed and multiplied together to get the DFT of the coefficients of the candidate polynomial. Exception: if the degree is less than 500, Matlab's command "poly" is used to combine them. This is the second most time consuming part of the total program. The coefficients are calculated from the candidate zeros with the "ifft" command and compared to the original coefficients. If they are the same number, then the search and polish possibly found them all. If the polishing process says the roots are close and a comparison in the unfactoring process says the coefficients are close, then these are the two criteria for success.

If the **grid search** and **polish stages** did not find all of the zeros, the original polynomial is deflated by removing those zeros found from it. That is done by dividing the DFT of the coefficients of the original polynomial, point-by-point, by the DFT associated with the zeros away from the unit circle which are found by the rootSpectrum program (deflation in the frequency domain). This quotient polynomial is further deflated (this time in the coefficient domain) by deconvolving it with the inverse FFT of the DFT associated with the zeros very near the unit circle. The resulting new quotient polynomial is now the original polynomial with all of the zeros found by the grid search and polish stages removed from it. This, hopefully low degree, polynomial is factored with the Matlab "roots" function and polished, first against the quotient polynomial, then the original polynomial. These new candidate zeros are added to the set found by grid search and

polish, their associated spectra multiplied by that found in the “unfactoring” function, then checked to see if there are enough and if their associated coefficients are close to the original coefficients. If the answer is yes to both, the factoring is finished. If there are too few zeros, deflation will be performed again. This is repeated until all of the zeros are found or until no new ones are found. If there are still too few, a new search grid is designed with a smaller cell size and started again in stage one.

The basic mathematics behind this program can be found [here](#).

Mathematical Principles for the LF Algorithm

Addendum gives the mathematical definitions and ideas used by the Lindsey-Fox algorithm

Addendum: Mathematical Principles

The mathematical principles at the core of the [Lindsey-Fox](#) algorithm for polynomial factoring are given here.

An N^{th} degree polynomial is denoted by

Equation:

$$P(z) = a_0 + a_1z + a_2z^2 + \cdots + a_Nz^N = \sum_n a_n z^n$$

or

Equation:

$$P(z) = \prod_k \left(z - z_k \right)$$

where $k = 1, 2, \cdots, N$ or

Equation:

$$P(z) = \prod_m \left(z - z_m \right)^{M_m}$$

where $m = 1, 2, \cdots, Q$ and $N = \sum_m M_m$. And a_n is the n^{th} coefficient, z_k is the k^{th} zero or root, N is the degree of the polynomial, M_m is the multiplicity of the m^{th} zero, and Q is number of distinct roots or zeros.

The **fundamental theorem of algebra** states that an N^{th} degree polynomial has N zeros.

The length- L discrete Fourier transform (DFT) of the N coefficients of a polynomial $P(z)$ with $L \geq N$ are the L equally spaced samples of the polynomial evaluated on the unit circle of the complex plane.

Equation:

$$DFT_L\{a_n\} = P\left(e^{2\pi ik/L}\right)$$

for $k = 0, 1, 2, \dots, L - 1$

If the coefficients are multiplied by a geometric sequence, r^n , the DFT of this modulated set of coefficients are the L equally spaced samples of the polynomial evaluated on a circle of radius r in the complex plane.

Equation:

$$DFT_L\{r^n a_n\} = P\left(re^{2\pi ik/L}\right)$$

for $k = 0, 1, 2, \dots, L - 1$

Using Horner's method, the number of multiplications and additions necessary to directly calculate N equally spaced values of a degree N polynomial on the unit circle is proportional to N^2 . If evaluated with the DFT, it is also proportional to N^2 . If evaluated with the FFT, it is proportional to $N \log(N)$.

If the roots of a polynomial are at z , the roots of the same polynomial with the sequence of coefficients reversed ("flipped"), are at $1/z$.

Equation:

$$P_{a'}(1/z) = P_a(z)$$

The "Minimum Modulus Theorem" can be stated several ways. A way most applicable to our test of the 3 node by 3 node cells is: If the minimum of an analytic function of a complex variable occurs in the interior of an open set, the minimum must in fact be a zero of the function.

If Newton's algorithm is applied to a polynomial and is started sufficiently close to a zero, it will quadratically converge to that zero if the zero is simple. If the zero is multiple, it still converges but only linearly. If Laguarre's algorithm is applied to a polynomial and is started sufficiently close to a zero, it will cubically converge to that zero if the zero is simple. If the zero is multiple, it still converges but only linearly.

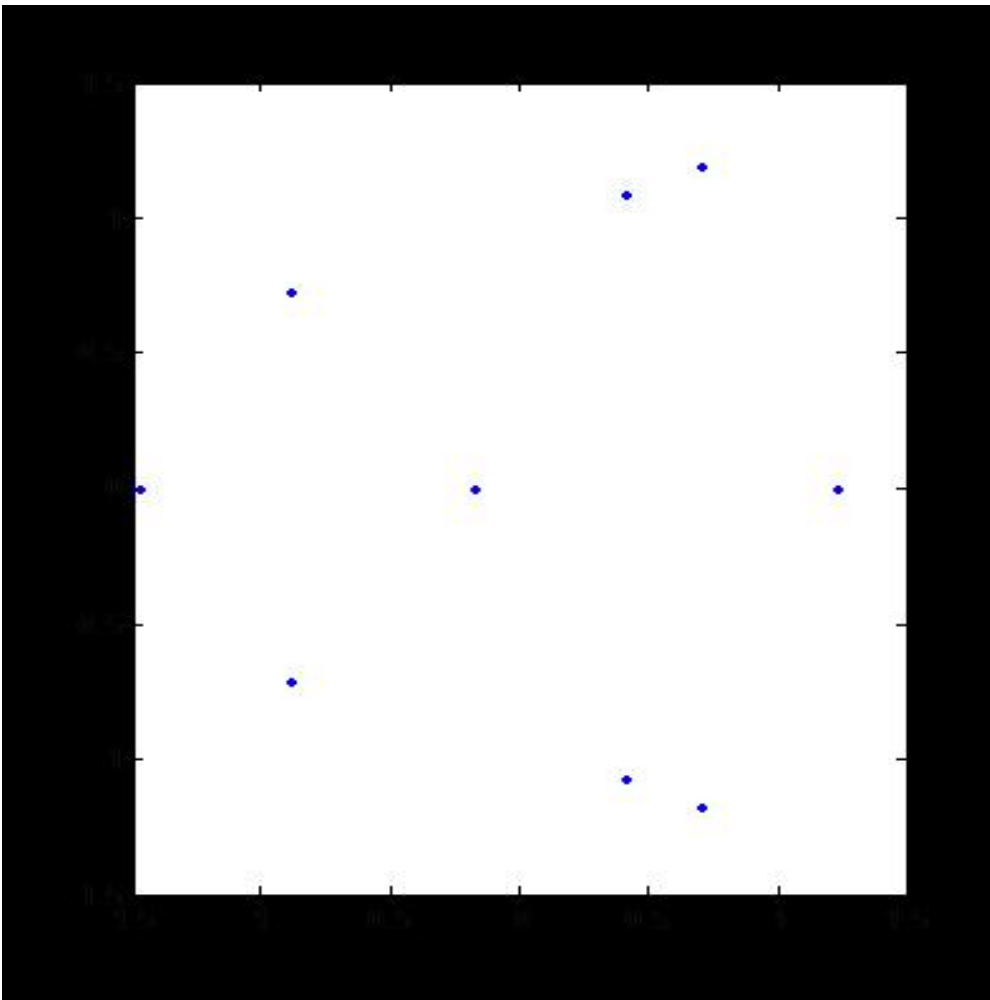
Zero location for Polynomials with Random Coefficients

The zeros of a polynomial with random coefficients cluster around the unit circle in the complex plane. As the degree increases, the roots cluster tighter and tighter to the unit circle. This phenomenon is illustrated here.

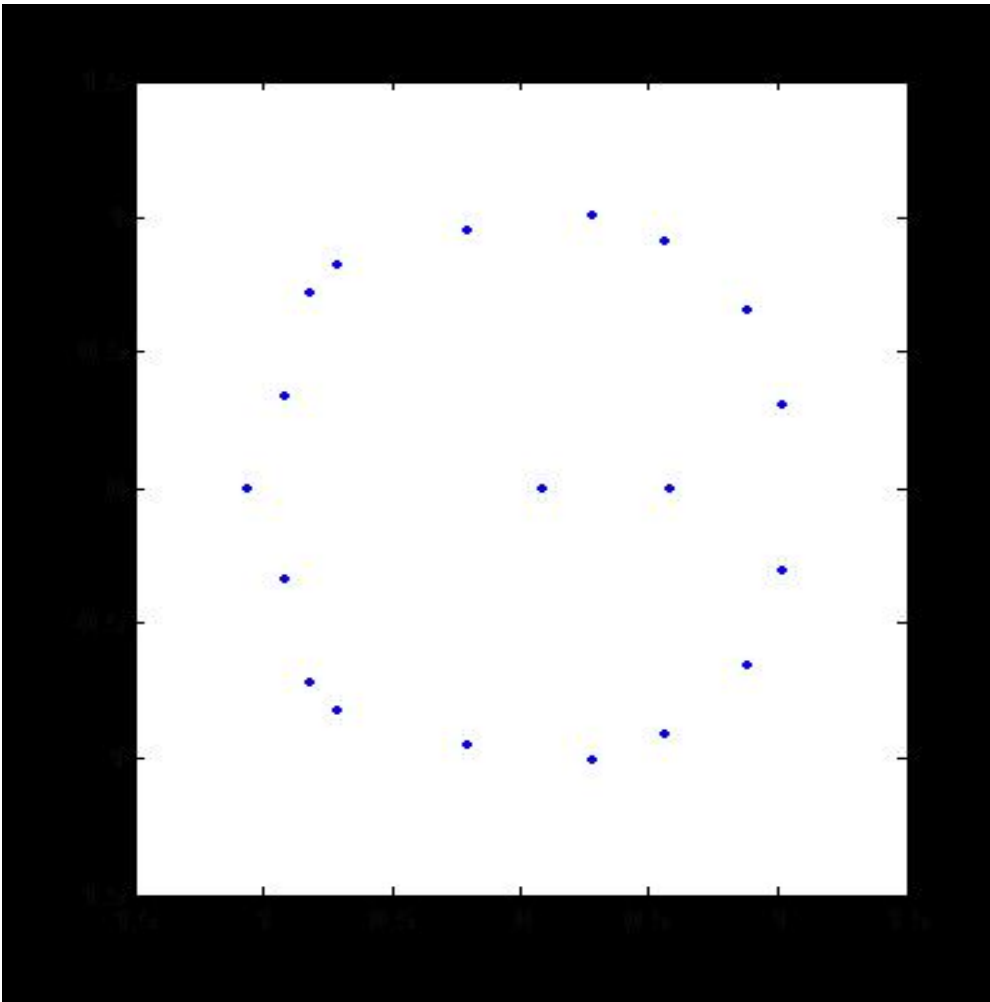
Zero Location of Polynomials with Random Coefficients

The location of the zeros of polynomials with random coefficients have the remarkable property of being clustered around the unit circle of the complex plane. They are uniformly distributed around the unit circle with a very sharp peak in their radial distribution at one. This has been reported and described in 1966 by Arnold and [others](#), but only with the development of the [Lindsey-Fox](#) algorithm have we been able to factor really high degree polynomials and observe the phenomenon. Thousands of multi-thousand degree polynomials with random coefficients have been factored and a few multi-million degree ones as well. The following plots of the locations of these zeros will illustrate the phenomenon. There is a [literature](#) on this subject, but seeing the actual root location is still impressive and informative and aids in developing other factoring strategies.

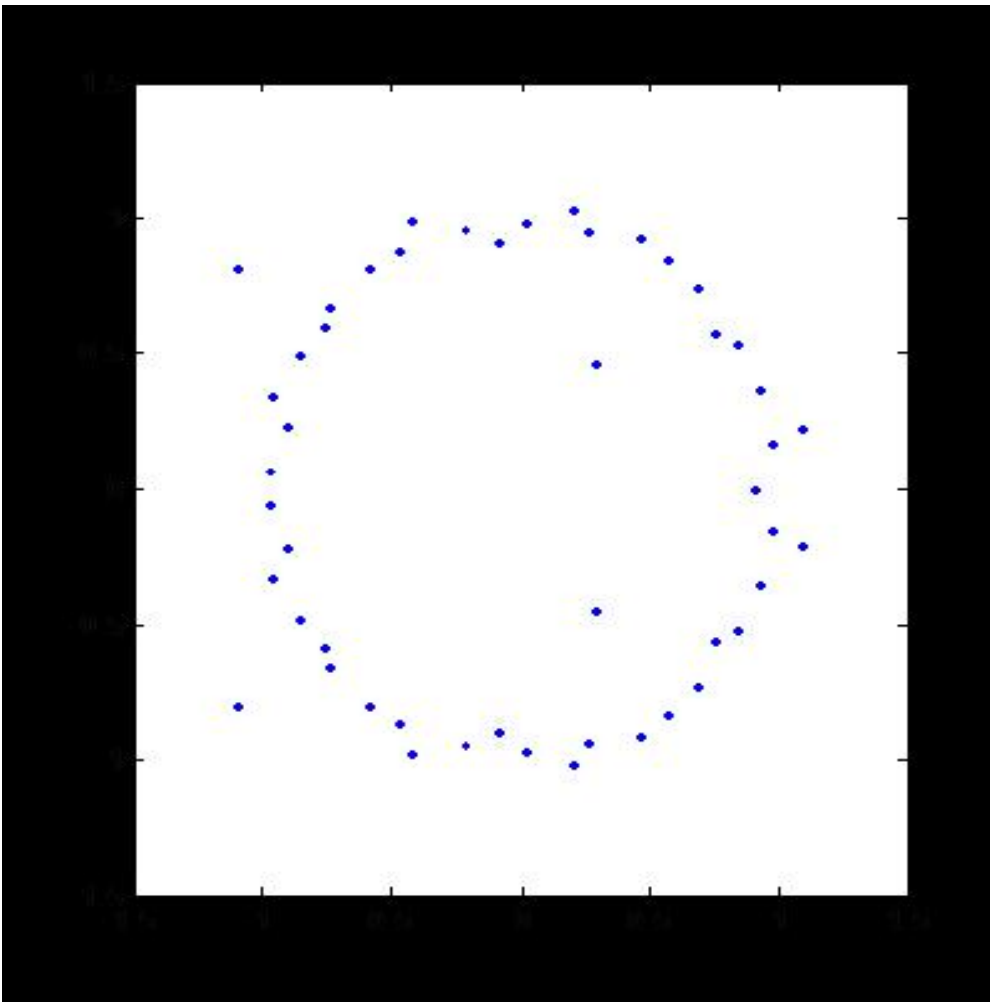
The following figures show the zero locations for random coefficient polynomials of degree 10 through 2000. These were found using Matlab by generating the random coefficients with the “rand” command, then factoring the resulting polynomial with the “roots” command (an eigenvalue method), and finally creating the graphics with the “plot” command. The results are relatively independent of the type of distribution of the random coefficients.



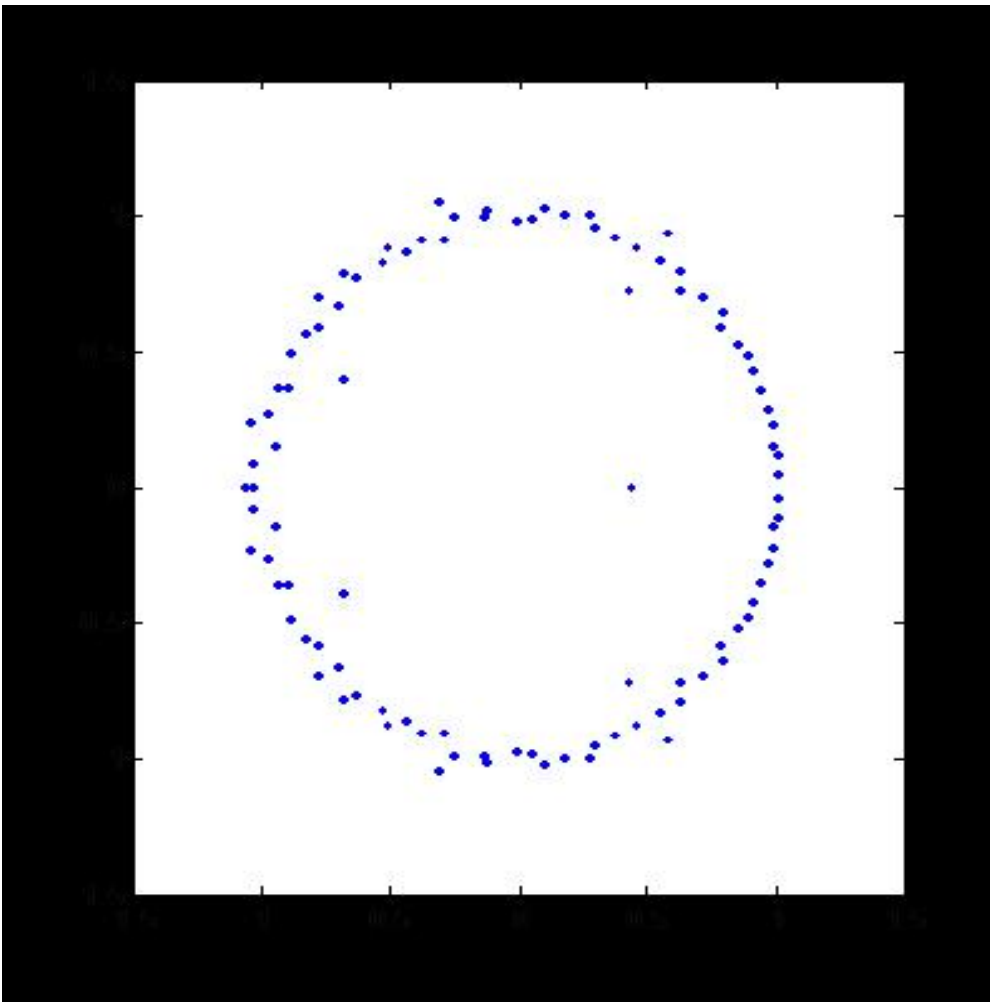
Zero Locations for $N=10$



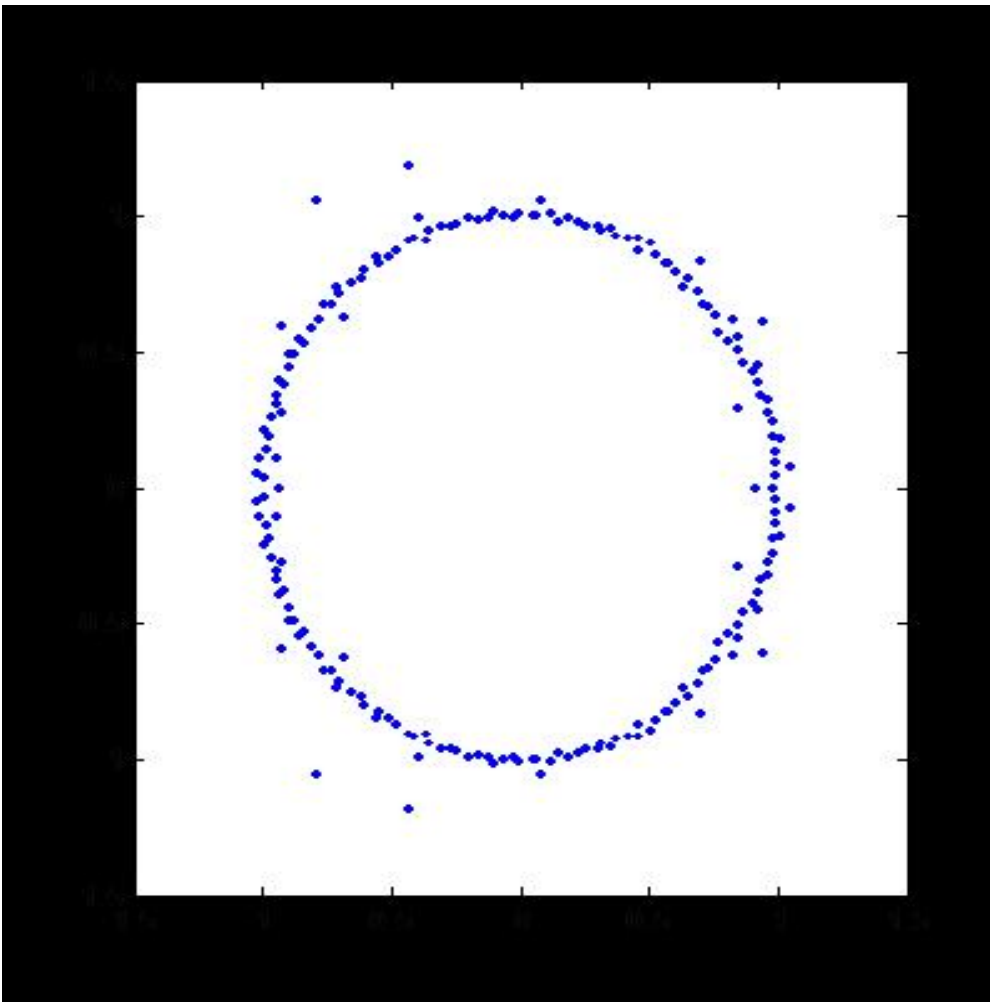
Zero Locations for $N=20$



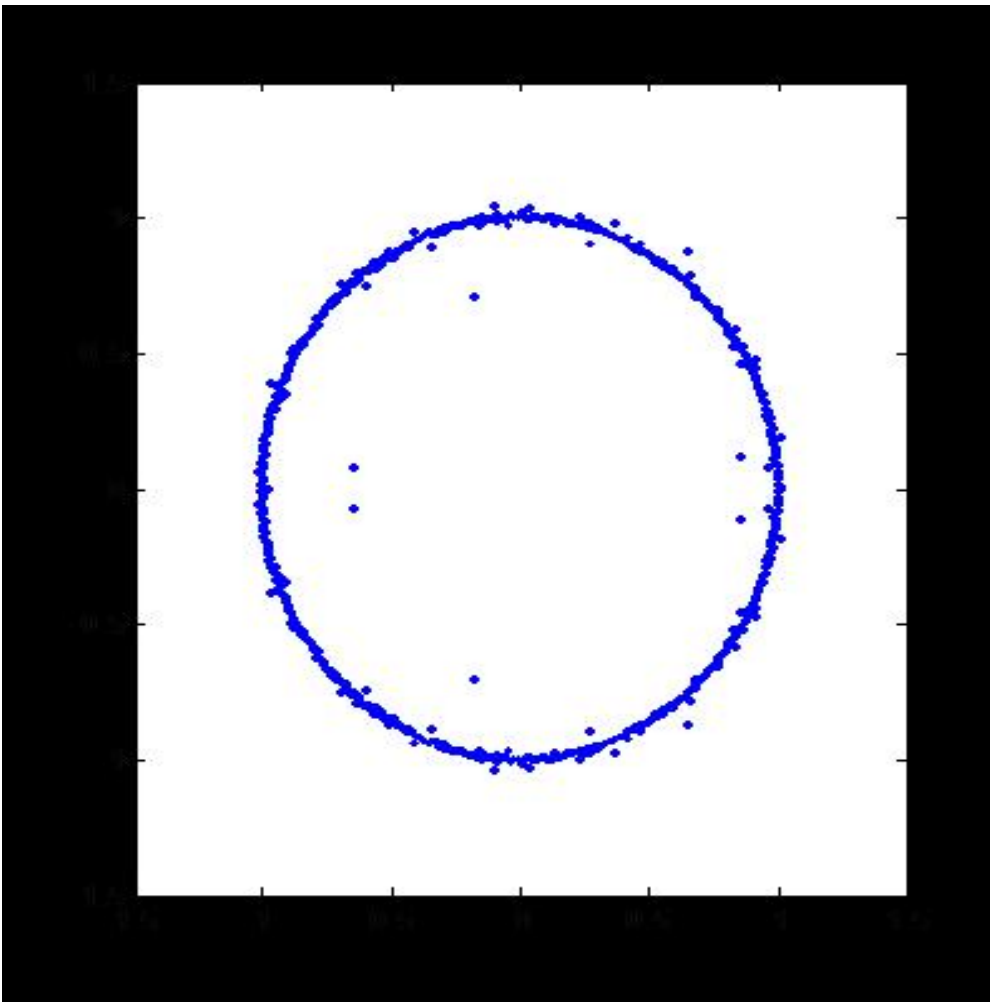
Zero Locations for $N=50$



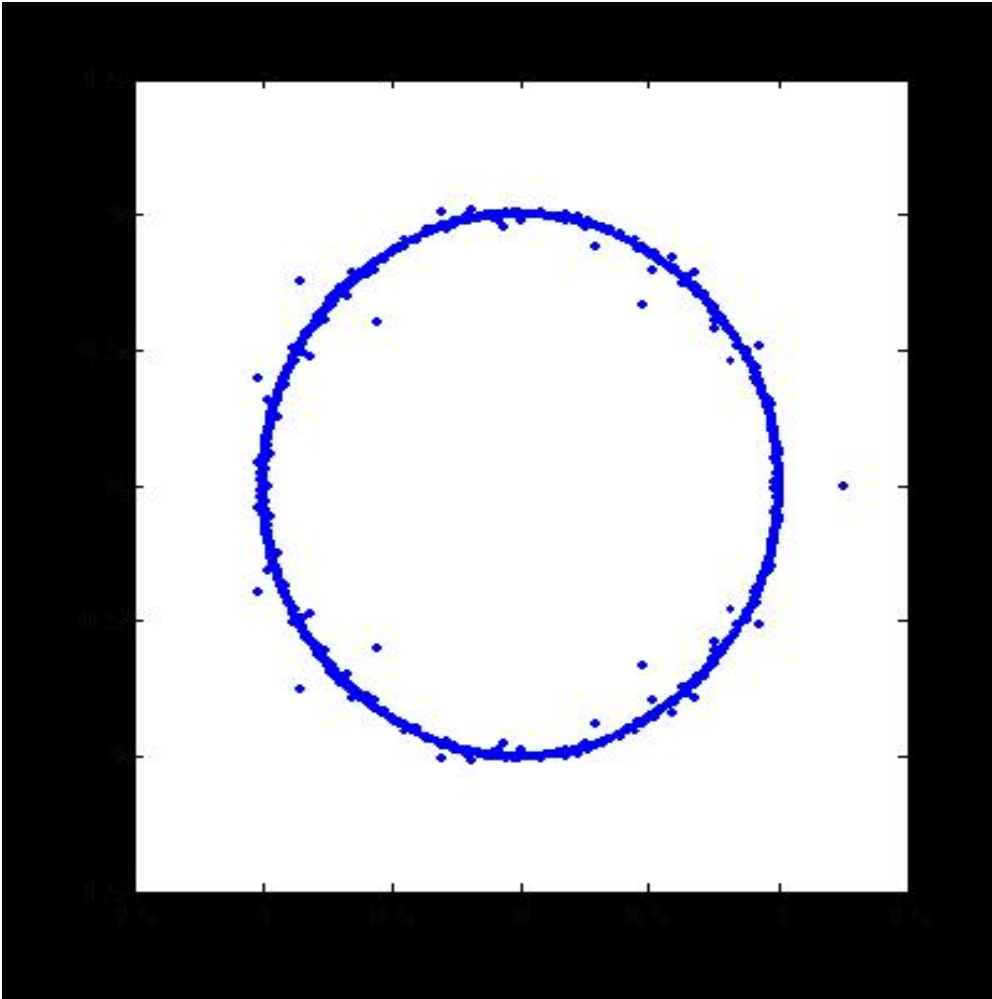
Zero Locations for $N=100$



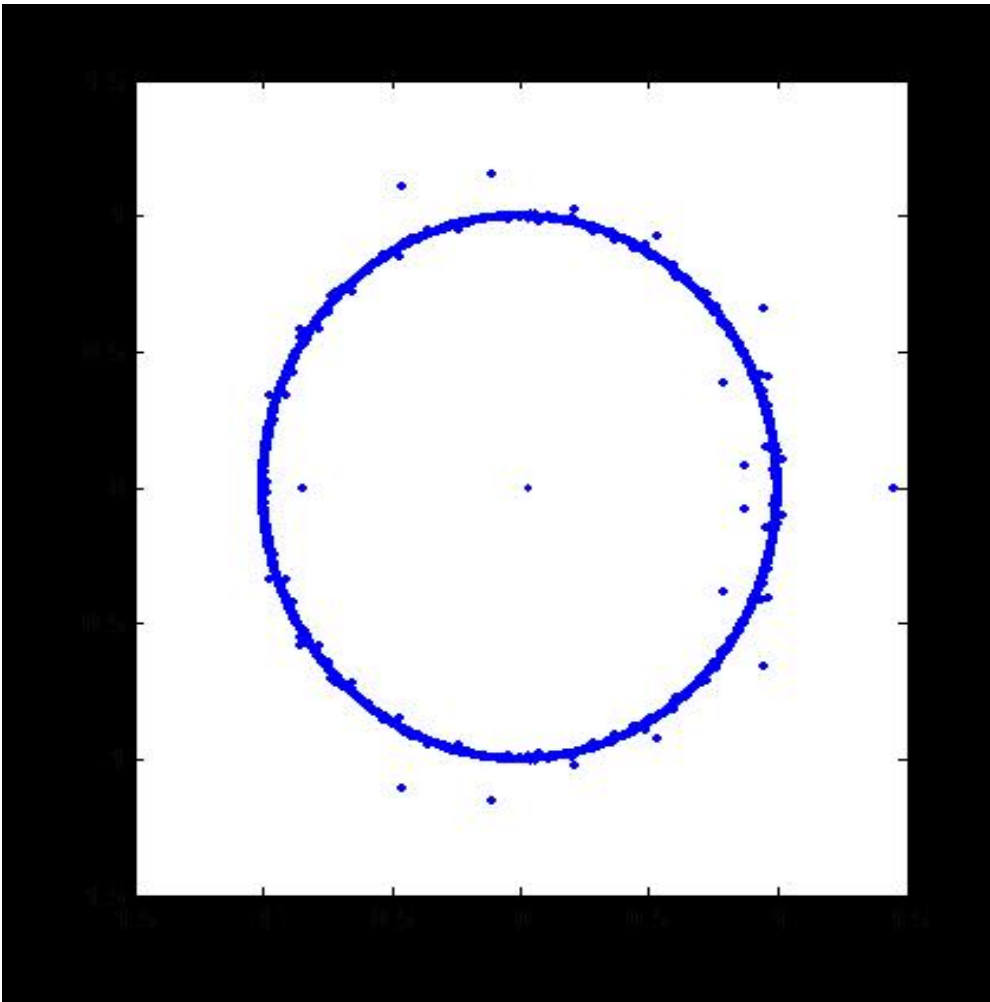
Zero Locations for $N=200$



Zero Locations for $N=500$



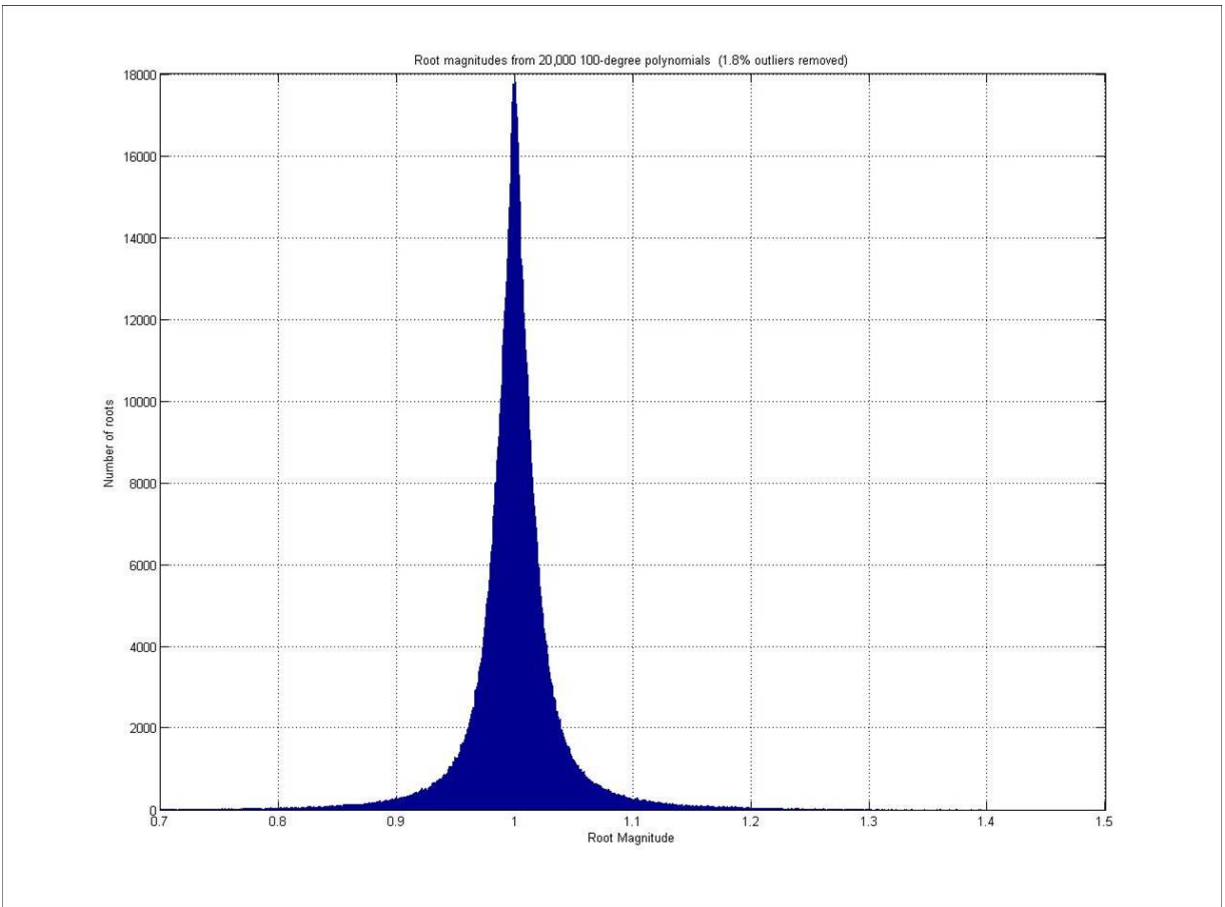
Zero Locations for $N=1000$



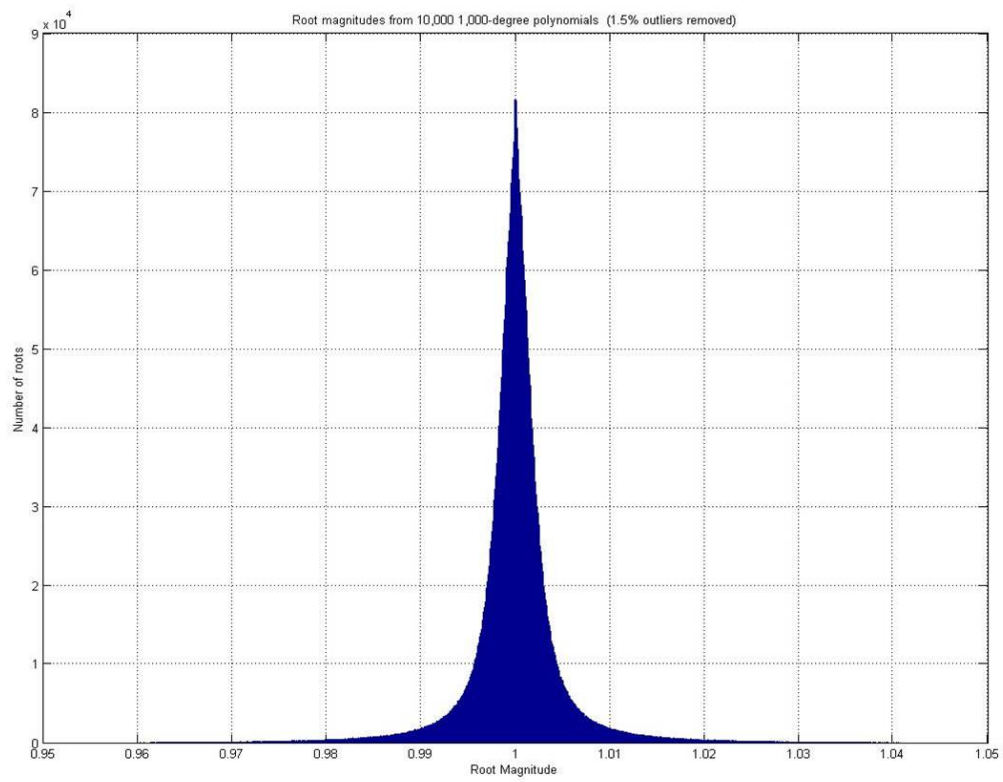
Zero Locations for $N=2000$

The following figures show histograms of the radial distribution or density of the zeros of polynomials with random coefficients. The random coefficients were generated in Matlab and the factoring was done with the [Lindsey-Fox algorithm](#) in Matlab.

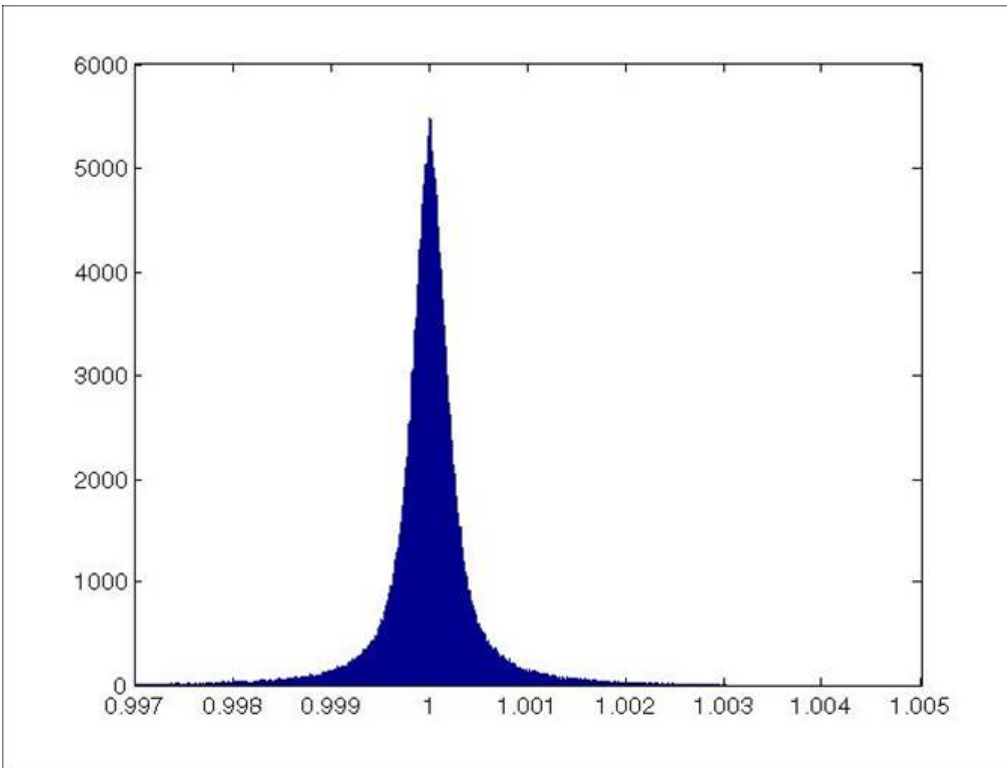
Although it is not easy to see from these figures, the shapes of the distributions are fairly independent of degree and the width is almost exactly linear with the inverse of the degree. As the degree goes to infinity, the zeros all go to the unit circle!



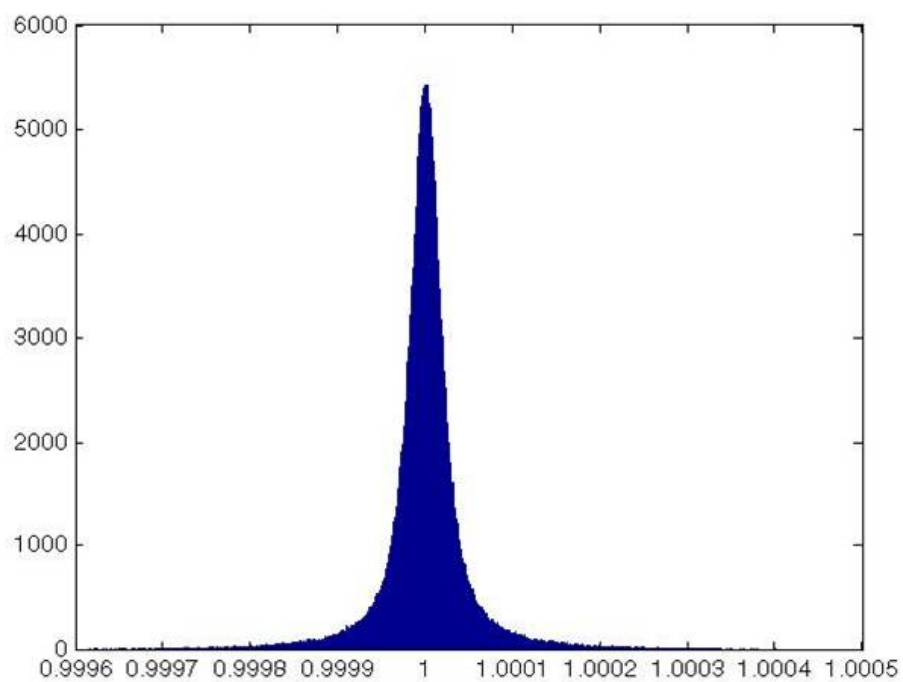
Number of Zeros at a Radial Distance in the Complex plane, $N=100$



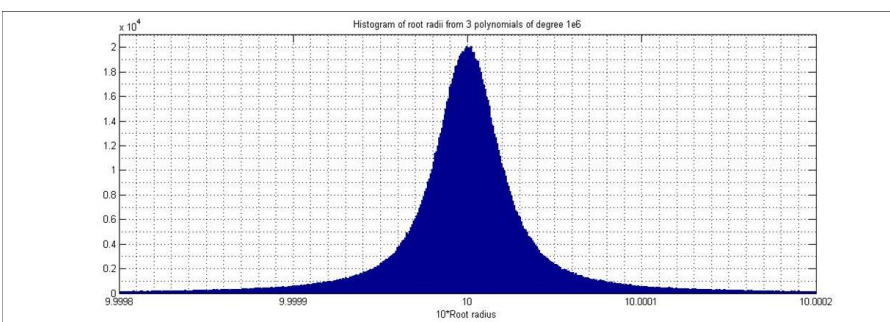
Number of Zeros at a Radial Distance in the Complex plane, $N=1,000$



Number of Zeros at a Radial Distance in the Complex plane, $N=10,000$



Number of Zeros at a Radial Distance in the Complex plane, $N=100,000$



Number of Zeros at a Radial Distance in the Complex plane, $N=1,000,000$